CS 640 Introduction to Computer Networks
Spring 2009
http://pages.cs.wisc.edu/~suman/courses/wiki/doku.php?id=640-spring2009

Programming Assignment 1: Mock Name Server (MNS)

Assigned: February 12, 2009
Due: February 19, 2009, 11:59pm

1 Introduction

The Internet's name service is used to resolve the mapping between a hostname and an IP address. There are two sets of entities that comprise the name service system – name servers and resolvers (clients). In programming assignment 0, you have implemented a simple version of resolvers. In this assignment, you will implement a simplified version of the name server.
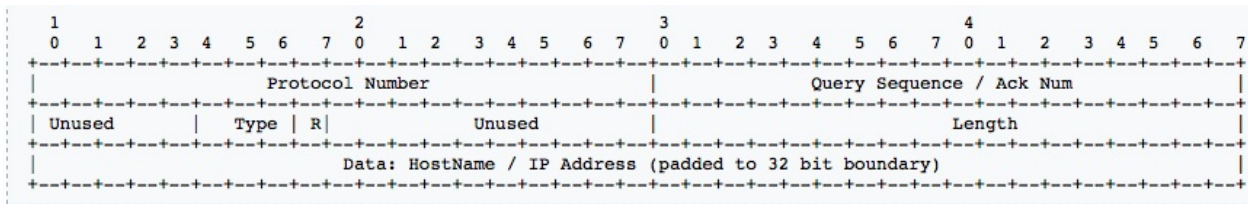
In your implementation, the server will run on a machine at a defined port number. The clients, called resolvers, will contact the server with queries about hostnames and IP addresses. On receiving a name service query, the server will lookup the query in its local database and respond with an appropriate answer. If the server does not know the answer, but knows of another server that may have the answer, it will send a redirect response to the resolver with the IP address and port number of that server. The resolver can then query the new server with the request. The server and the client (resolver) will use UDP – the User Datagram Protocol – to communicate with each other.

2 Protocol Specification

The protocol works as follows. The client will construct queries of different types and send them to the server. The server will query its internal database for each client query and send back a response. Depending upon the response, the client either outputs the response and terminates or queries another server. If a client does not get a response back from the server within a given timeout period, it assumes that the server is unreachable, outputs an appropriate error message, and terminates.

Packet Structure

Both the query and the response packets have the same header structure as shown below. (Refer to FAQ on how to set the fields)

```
       1                   2                   3                   4
 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|              Protocol Number          |        Query Sequence / Ack Num        |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
| Unused      |   Type  | R|       Unused          |              Length          |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|              Data: HostName / IP Address (padded to 32 bit boundary)           |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
```

• **Protocol Identifier**: (2 bytes) Set to the integer value 640.

• **Query Sequence/Ack Number**: (2 bytes) The value of this sequence number is assigned by the client and is passed as an argument to the client program. On its response, the server increments it by one.

• **Type**: (3 bits) This indicates the type of the message.
  000 indicates IP address to hostname resolution request (sent by the client).
  001 indicates hostname to IP address resolution request (sent by the client).
  010 indicates IP address to hostname resolution response (sent by the server).
  011 indicates hostname to IP address resolution response (sent by the server).
  100 indicates that the server could not handle this request (sent by the server).
  101 indicates a "redirect" response from the server (sent by the server).
  110 indicates that the server should terminate. Note that unless the server receives this message, the server should not terminate. (sent by the client to the server). Note that it is only in this assignment that the clients can terminate the server. In the real Internet's name service the clients do not have this privilege.

• **R**: (1 bit) This is reserved for future use and can be either 0 or 1 in the messages. It is ignored by both the client and the server.

• **Unused**: These bits are ignored by the client and the server.

• **Length**: (2 bytes) The length of the packet (in bytes) including the header sent by the client and the server.

• **Data**: (Variable length) Depending upon the packet type, this field can have an IP address (000, 011), hostname (001, 010) or IP address and port number (101).
  (1). For an hostname to IP address resolution request, this field contains the IP address;
      (2). For an IP address to hostname resolution response, this field contains a hostname;
      (3). For a redirect response it contains the IP address and port number of the server that should be contacted next by the resolver and zero-padding.
      (4). The IP address is stored in the network format as a 32-bit integer (see inet_ntoa() and inet_aton()).
      (5). The port is stored in the network format as a 16-bit integer (see htons() and ntohs()).
      (6). A hostname is stored using a NULL terminated text string, padded to a 32-bit word boundary (see FAQ for further discussion)

• **Note**: the numbers such as protocol, sequence number, length, and port in the packet data must be network order. It means that you must use htons() function to do the transformation before you pack the number into the packet. The IP number is already in network order and no transformation is needed.

3 Server/Client Details

The server will receive packets on a pre-defined port number. Both the server and the client will silently discard any packet which does not meet the specifications defined above. In particular, if the client receives a malformed packet it will discard this packet and continue to wait for a correctly formatted packet from the server, or until the timeout period is reached. The database for each server will comprise of 2 ASCII files

> • **Mapping file**: It will contain the IP address to hostname mapping in the format shown below. The first line will contain the total number of entries in the file. Each subsequent line of this file will contain a hostname and corresponding IP address separated by white space. For example,

> 2
> machine1.cs.wisc.edu 128.45.6.3
> machine2.yahoo.com 12.4.3.7

> (You can assume that there will be no errors in the data format for this file.)

> • **Redirect file**: It will contain the list of hostnames and IP addresses for with the current server does not know the answer but knows of another server which can possibly have an answer. The first line will contain the total number of entries in the file. Each subsequent line of this file will contain a hostname/IP address and the corresponding IP address and port pair at which a server should be queried next. The format of the file is shown below.

> 4
> machine3.cs.wisc.edu 128.45.6.10 2345
> 128.45.6.4 128.45.6.11 2346
> machine4.cs.wisc.edu 128.45.6.10 2348
> 128.45.6.5 128.45.6.10 2346

> (You can assume that there will be no errors in the data format for this file.)

Upon receiving a valid query packet, the server will first query its Mapping file for the information requested. If it fails to find the information there, it will next query its Redirect file to find the server that might have the requested information. If it finds an appropriate server, it will respond to the client with a redirect message. Note that the server will construct an appropriate response packet with its answer. For queries that cannot be answered either with a resolution response or with a redirection message, the data part will be zero bytes and the packet

type will be set to 100 (binary) as described above. For each query, the client uses a sequence number value which is passed to it as an argument. For a query with sequence number x, the server will send its response with an acknowledgment number value set to x+1. The client should verify that this is true in the response message that it receives. If the server does not respond within some time period (specified by a timeout argument) the client will output an appropriate error message and terminate. The server terminates on receiving a client request with the type field set to 110. The server does not send a response to the client in this case. The client that sends this request should also terminate immediately after sending this message.

4 Executables

The following is the expected executable format for the server (along with the necessary arguments):

• **Server**

server -p <port> -f <mapping-file-name> -r <redirect-file-name>

> port: Port number of the host at which the server will run.
> mapping-file-name: Name of the Mapping file to use as described above.
> redirect-file-name: Name of the Redirect file to be used as described above.

We will run the server executable by providing the mapping/redirect files. The server is not required to output any text to screen in this programming assignment. You should take care that your code is robust and handles possible erroneous messages from the client.

5 Output

We will test the server with a client implementation which follows the above specifications. If you have implemented the server correctly, the client will output the following:

• If the server is provided with a mapping-file which has the answer to the query:
  • For IP address to hostname resolution request (req-type 0) the client should output the hostname in a single line. For example:
    machine.cs.wisc.edu
  • For hostname to IP address resolution request (req-type 1) the client should output the IP address in a single line. For example:
    128.56.49.12
• If the server is provided with a mapping-file which does not have the answer to the query:

• If the server has a redirect-file which has the answer to the query, it should respond with type 101 and appropriate data. The client will then output the IP address and port number of the new server which has the mapping
• If the server has a redirect-file which does not have the answer to the query, it must respond with the type field set to 100 (binary), i.e. the server could not resolve the request. The client should output following text in a single line:
  unresolved

• For request type 6 (termination request), the server should shut down.